A Lagrangian Approach to Dynamic Interfaces through Kinetic Triangulation of the Ambient Space

Jean-Philippe Pons, Jean-Daniel Boissonnat

Volume 26 (2007), number 2 pp. 227–239

COMPUTER GRAPHICS *forum*

A Lagrangian Approach to Dynamic Interfaces through Kinetic Triangulation of the Ambient Space

J.-P. Pons[1] and J.-D. Boissonnat[2]

[1] École Nationale des Ponts et Chaussées, Marne-la-Vallée, France
[2] INRIA, Sophia-Antipolis, France

Abstract

In this paper, we propose a robust and efficient Lagrangian approach for modeling dynamic interfaces between different materials undergoing large deformations and topology changes, in two dimensions. Our work brings an interesting alternative to popular techniques such as the level set method and the particle level set method, for two-dimensional and axisymmetric simulations. The principle of our approach is to maintain a two-dimensional triangulation which embeds the one-dimensional polygonal description of the interfaces. Topology changes can then be detected as inversions of the faces of this triangulation. Each triangular face is labeled with the type of material it contains. The connectivity of the triangulation and the labels of the faces are updated consistently during deformation, within a neat framework developed in computational geometry: kinetic data structures. Thanks to the exact computation paradigm, the reliability of our algorithm, even in difficult situations such as shocks and topology changes, can be certified. We demonstrate the applicability and the efficiency of our approach with a series of numerical experiments in two dimensions. Finally, we discuss the feasibility of an extension to three dimensions.

Keywords: curve, interface, deformation, Lagrangian, topology, collision, triangulation, kinetic data structure, exact computation, multiple junction

ACM CCS: I.3.5 Computer Graphics: *Computational Geometry and Object Modeling – Curve, surface, solid, and object representations*, 6.8 Simulation and Modeling *Types of simulation – Animation*.

1. Introduction

Modeling dynamic interfaces between several materials undergoing large deformations is a ubiquitous task in science and engineering. The existing techniques roughly fall into two categories: Eulerian and Lagrangian formulations. It is commonly admitted that both viewpoints have strengths and weaknesses and that only a hybrid approach can overcome the limitations of both. In this paper, we propose a method which has the particularity of being purely Lagrangian and of retaining all the associated advantages, while achieving topological adaptivity with a comparable robustness and efficiency to Eulerian methods.

1.1. Eulerian methods

The Eulerian formulation casts deformation as a time variation of quantities defined over a fixed grid. The interfaces have to be represented implicitly, since the grid does not conform to them. In computational physics, this is also known as the *front capturing* method. Two notable front capturing techniques are the *level set method*, introduced by Osher and Sethian [OS88], and the *volume-of-fluid (VOF) method*, pioneered by Hirt and Nichols [HN81].

Hereafter, we will mainly focus on the level set method because it is an established technique in fluid animation. Basically, this method consists in representing the interface as the zero level set of a higher-dimensional scalar function. The movement of the interface can be cast as an evolution of the embedding level set function by an Eulerian PDE (partial differential equation). We refer the reader to some good reviews [Set99, OF01] for all the details about the theory, the recent developments, the implementation and the applications of the level set method.

Submitted September, 2006
Revised February, 2007
Accepted February, 2007

On the one hand, this approach has several advantages over an explicit Lagrangian representation of the interface: no parameterization is needed, topology changes are handled automatically, intrinsic geometric properties such as normal or curvature can be computed easily from the level set function. Last but not least, the theory of *viscosity solutions* provides robust numerical schemes and strong mathematical results to deal with the evolution PDE. These advantages explain the popularity of the level set method for multi-phase fluid flow simulation [SSO94] in CFD (computational fluid dynamics), as well as for computer animation of fluids with free surfaces [FF01; EFFM02; EMF02; LGF04; ELF05].

On the other hand, several serious shortcomings limit the applicability of the level set method. *First*, the higher dimensional embedding makes the level set method much more expensive computationally than explicit representations. Much effort has been done to alleviate this drawback, leading to the *narrow band* methodology [AS95] and to the *PDE-based fast local level set method* [PMO*99]. More recently, octree decompositions have been proposed [LGF04; ELF05; LFO06; COQ06] to circumvent the typically fixed uniform sampling of the level set method, in order to reach high resolution (typically an effective resolution of 512^3) while keeping the computational and memory cost sustainable. However, these methods somewhat lose the simplicity of the original level set method, as an efficient implementation of such tree-based methods turns out to be a tricky task.

Second, as discussed and numerically demonstrated by Enright *et al.* in [EFFM02], the level set method is strongly affected by mass loss, smearing of high curvature regions and inability to resolve very thin parts. These limitations have motivated the development of some hybrid Eulerian-Lagrangian methods, such as the *particle level set method* outlined by Foster and Fedkiw [FF01] and later improved by Enright and coworkers [EFFM02; EMF02; ELF05]. While the latter method yields state-of-the-art results, an objection could be the large number of parameters controlling the particle reseeding strategy included in this approach. To some extent, also related is the *contouring method* [Str01; SJ02; BGOS06], which converts back and forth from a Lagrangian mesh to an implicit function evaluated on a regular Eulerian grid using distance computations and isocontour extraction algorithms.

Third, purely Eulerian formulation is not very appropriate for tracking interface properties such as color or texture coordinates, as may be needed in computer graphics applications. An approach based on a coupled system of Eulerian PDEs was recently proposed by Pons *et al.* to overcome this limitation within the level set framework [PHKF06], but this capability comes at a significant additional computational cost.

1.2. Previous Lagrangian methods

The Lagrangian formulation adopts a more "natural" point of view. It explicitly tracks the interfaces between the different materials with some points advected by the motion. There are mainly two classes of Lagrangian techniques: *mesh-based* methods and *particle-based* methods. We first tackle the mesh-based approach, also known as the *front tracking* method in computational physics [TBE*01, UT92]. When dealing with large deformations, this approach is hampered by distortion and entanglement of the mesh, source of numerical instabilities or even of breakdowns of the simulation. For instance, if corners and cusps develop in the evolving front, front tracking methods usually form "swallowtail" solutions. These defective parts of the interface must be detected, then removed through intricate delooping procedures.

Another major shortcoming of the mesh-based Lagrangian approach is that a fully automatic, robust and efficient handling of topology changes remains an open issue, despite many heuristic solutions proposed in various fields of application (e.g. [MT99, MT00; LM99] in medical imaging, [TBE*01, SJ02] in computational fluid dynamics, . . .).

McInerney and Terzopoulos [MT99, MT00] propose topology adaptative deformable curves and meshes, called *T-snakes* and *T-surfaces*. During the evolution, the model is periodically resampled by computing its intersections with a regular simplicial decomposition of space. A labeling of the vertices of the simplicial grid as inside or outside of the model is maintained. Besides being computationally expensive, this procedure loses the desirable adaptivity of the Lagrangian formulation, by imposing a fixed uniform spatial resolution.

Lachaud and Montanvert [LT05, LM99] use the concept of δ-triangulation. A length parameter δ is used to control the sampling of the mesh and to detect self-intersections, by monitoring the distance between pairs of neighbor and non-neighbor vertices. Besides being only approximate, this approach for topological flexibility is costly, even when optimizing the pairwise distance computations with an octree structure.

This major shortcoming of mesh-based methods have gained popularity to the particle-based approach [TPF89; DG95; FM96; DC96; CD97; MCG03; PTB*03; MKN*04; PKA*05] for representing dynamic interfaces undergoing complex topology changes. However, it is generally admitted that with the meshless approach, the localization of the interface and the computation of interface properties such as normal and curvature gets cumbersome.

For sake of completeness, let us also mention a recent approach for fluid simulation based on dynamic meshes [KFCO06]. This method bears some resemblance with ours, in the sense that it maintains a triangulation of the computational domain during motion, which conforms to a moving object. However, their purpose, fluid/rigid body coupling, is different from ours, fluids with free interfaces: in their case, the geometry and the topology of the front is fixed.

1.3. Novelty of our method

In this paper, we present a purely Lagrangian approach, which combines the advantages of front tracking and front capturing methods, while discarding their respective drawbacks. Our work brings a robust and efficient solution to remeshing and topological adaptivity for Lagrangian dynamic interfaces in two dimensions. The principle of our approach is to maintain a two-dimensional triangulation of space which embeds the one-dimensional polygonal description of the interfaces. Interestingly, this increase of dimension bears similarity to the spirit of the level set method. Topology changes can then be detected as inversions of the triangular faces (called faces for short) of the triangulation. Each face is labeled with the type of material it contains. This embedding triangulation enforces directly watertight interfaces free of loops, swallowtails or self intersections at all times.

The connectivity of the triangulation and the labels of the faces are updated consistently during deformation, within the *kinetic data structures* framework, introduced by Basch *et al.* [BGH99]. Rather than trying to repair the possibly entangled triangulation after each iteration of the simulation, we smoothly interpolate the coordinates of interface points between two consecutive time steps, and we modify the connectivity of the triangulation and the labels of the faces exactly as and when it is required. As a result, our *lazy kinetic triangulation* has the desirable property not to introduce any unnecessary perturbation of the interfaces in the absence of topology changes, and to handle them exactly and efficiently when they occur.

Being purely Lagrangian, our method does not suffer from mass loss which plagues the level set method, and can track material properties such as color or texture coordinates during motion at no additional cost, while being free of the localization problem of particle-based approaches.

Moreover, our method is not limited to two-phase motion. It seamlessly accommodates any number of materials, whereas this requires special care in most existing Lagrangian and Eulerian methods (*cf* for example [SSC02] on the treatment of triple junctions with the level set method).

The remainder of this paper is organized as follows. Section 2 gives some background on kinetic data structures, and particularly on kinetic triangulations and their application to efficient collision detection. The different elements of our lazy kinetic triangulation are described in Section 3. In Section 4, we give the details of our implementation with CGAL (Computational Geometry Algorithms Library) [BDTY00] and we demonstrate the applicability and the efficiency of our approach with a series of numerical experiments. Finally, we discuss an extension to three dimensions in Section 5.

2. Background

2.1. Kinetic data structures

Kinetic data structures, introduced by Basch *et al.* [BGH99], are a general framework to efficiently maintain a discrete attribute of a set of moving objects in time. Typically, this attribute consists of a combinatorial description such as the convex hull [BGH99] or the Delaunay triangulation [AGMR98, GR04] of a set of moving points.

This framework was motivated by the important limitations of the method of widest practical use, which is a naive incremental update of the attribute at some predefined (usually uniformly distributed) time samples. Because the distribution in time of the actual changes of the attribute is far from uniform, the choice of the time step is problematic. Any time step is bound to oversample the system sometimes, then performing useless computations, and to undersample it sometimes, then missing some possibly important intermediate states of the attribute.

In contrast, kinetic data structures take advantage of the knowledge of the motion to exactly determine the times when *events* occur, i.e. when the attribute changes. No computation is made between two consecutive events. This is achieved by maintaining a set of elementary geometric predicates, which is logically equivalent to the current state of the attribute. For instance, the validity of a 2D kinetic Delaunay triangulation [AGMR98, GR04] is certified by one *InCircle* predicate for each triangle, plus some *Orientation* predicates for the edges of the convex hull. The next change of the attribute is then scheduled at the earliest time for which one of the predicates fails. This is possible in practice whenever the trajectories of the objects are described by low-degree polynomials, so that the failure times of the predicates are roots of polynomials. Moreover, the efficient processing of the events in chronological order is achieved by storing the failure times in a global priority queue. When an event occurs, the attribute is properly modified, then some predicates affected by the change are created, rescheduled or deleted, while the priority queue is updated accordingly.

We refer the reader to some recent reviews [AGE*02, Gui04] for a thorough description of the kinetic data structures framework and its applications to the maintenance of classical combinatorial structures such as convex hulls, Voronoi diagrams and Delaunay triangulations, closest pairs, minimum spanning trees, etc.

2.2. Kinetic triangulations

Maintaining a triangulation of a set of moving points is of particular interest for our problematic of modeling materials undergoing large deformations and topology changes. There has been a significant amount of work on time-dependent triangulations within the kinetic data structures framework, but most of it was focused on algorithmic complexity.

An important result in this area was obtained by Agarwal *et al.* [ABdB*99]: maintaining a triangulation with a kinetic data structure requires at least a quadratic number of events in the worst case, even if the addition of a linear number of Steiner (i.e. additional) points is allowed. So far, no algorithm was shown to attain this lower-bound result. The algorithm described in [ABG*02] processes $O(N^{7/3})$ events in the worst case. More recently, in [AWY04], an algorithm was proposed that only requires $N^2 2^{O(\sqrt{\log N \log\log N})}$ events.

In the particular case of the Delaunay triangulation, the gap between theoretical bounds and existing algorithms is wider, with a quadratic best lower bound (i.e. the same as for an arbitrary triangulation) and a nearly cubic best known upper bound [AGMR98].

The spirit of our work significantly differs from that of the above references. We have several reasons to believe that the quadratic lower bound reported in [ABdB*99] is not relevant to our problem. First, our context is far from the worst-case scenario, since the boundary of the materials where the points are distributed, as well as the motion, typically exhibit some regularity properties. In [ABL03], it was shown that such a smoothness assumption reduces the complexity of the static 3D Delaunay triangulation of N points from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. We expect a similar complexity drop for a kinetic triangulation too, although we are not aware of a theoretical result in this direction.

Second, we do not resort to a canonically defined triangulation, such as a Delaunay triangulation [AGMR98, GR04] or a constrained fan triangulation [AWY04]. In other words, the state of our kinetic triangulation does not depend only on the positions of the points, but also on the history of the motion. As a consequence, it is very difficult to obtain formal statements on the complexity of our approach.

Last, our kinetic triangulation has the particularity of being coupled with a numerical simulation which computes the positions of the points at evenly distributed time instants. At each time step, all the trajectories are updated and the whole set of predicates is recomputed. With reasonably small time steps, few events occur during an interval of the simulation. Under these conditions, the computational expense of our kinetic data structure is mostly related to its *compactness*, as defined in [AGE*02, Gui04], i.e. the number of predicates used to certify the triangulation. It is exactly $N - 2$, the number of triangles, with our simple algorithm. In the following, we will discuss possible improvements on this number.

2.3. Collision detection

An efficient and exact detection of collisions between several moving polygonal objects can be achieved by maintaining a tiling of the free space between them with a kinetic data structure. A triangulation is the more immediate candidate for such a tiling. However, several authors advocate the use of a more complex structure, namely a pseudo-triangulation [BEG*99; KS02; ABG*02], that is to say a tiling of space with pseudo-triangles. A pseudo-triangle is a simple polygon composed of three concave chains joining at their endpoints. A kinetic pseudo-triangulation is much harder to implement than a kinetic triangulation, but has significant advantages in some situations.

Compared to a kinetic triangulation, a kinetic pseudo-triangulation requires a smaller number of elements and adapts to the free space between moving objects with fewer events. Also, it is very *compact*, as defined previously, under some assumptions on the type of motion, on the separation between the objects or on their convexity. However, these methods do not apply to non-convex objects under non-rigid motion, with the exception of [ABG*02]. Unfortunately, the latter approach does not guarantee a better compactness than our simple kinetic triangulation in the case of non-rigid motion, which makes the choice of a kinetic pseudo-triangulation questionable for our application.

Also, the reader should note that the problem that we address in this paper is less restrictive than collision detection: our method allows the simulation to continue after a collision, by properly handling the fusion or the splitting of the different materials.

3. Methods

The principle of our approach is to couple the numerical solver which outputs the position of the vertices at some (typically evenly-spaced) time instants, with a kinetic triangulation. The numerical solver is specific to each application and is not discussed here. Some examples will be briefly presented in Section 4. In this section, we describe the particularity of our *lazy kinetic triangulation* and explain its pertinence for the modeling of dynamic interfaces.

3.1. Embedding triangulation

A determinant feature of the triangulation we maintain is that a physical interpretation is attached not only to its vertices, but also to its faces, namely the type of material that it contains. For instance, in a multi-phase fluid simulation, each triangle is mapped to one of the phases. This information, that we call for short the *label* of a triangle, can be conveniently represented by an integer value. The interfaces of interest are embedded in this triangulation. They are composed of the edges adjacent to two triangles having different labels.

This representation has many virtues. It automatically enforces watertight interfaces. It seamlessly accommodates any number of materials. It makes the request of the type of material at some point very efficient. An example of this embedding triangulation, including three materials with some triple junctions, is displayed in Figure 1.

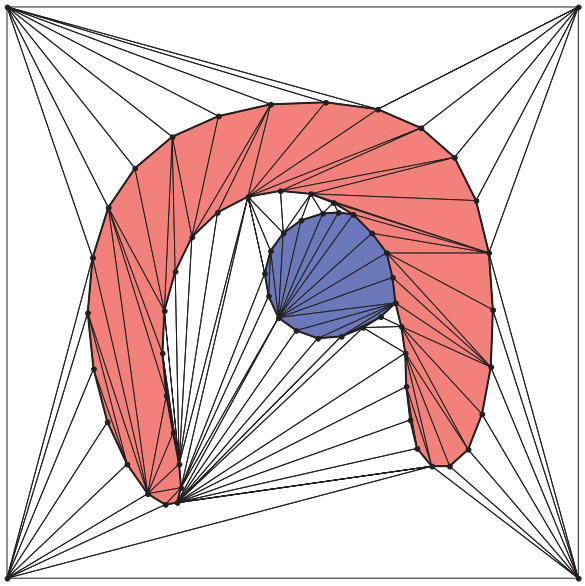

Figure 1: *Embedding labeled triangulation.*

3.2. Lazy approach

We now describe how the connectivity of the triangulation and the labels of the faces are consistently updated during motion, in order to prevent the breakdown of the simulation, and to faithfully reflect the deformations and the topology changes of the different materials.

Our kinetic triangulation is termed "lazy" because it undergoes no connectivity change as long as it remains a valid geometric triangulation, that is to say as long as the orientations of the triangles do not change and the triangulation covers the convex hull of the vertices. In other words, the triangulation is modified only when a triangle becomes flat or two consecutive border edges become collinear. The corresponding set of predicates is an *Orientation* predicate for each triangle, plus an *Orientation* predicate for each triplet of consecutive border vertices. The events are processed with elementary local connectivity modifications, namely edge flips and edge collapses, that we will describe in detail later.

Contrarily to a Delaunay triangulation, the state of our triangulation is not canonically defined: it does not depend only on the positions of the points, but also on the history of the motion. As a result, it does not suffer from an instability and a multiplication of meaningless events when the points are very close to a configuration change, like it happens with Delaunay when four or more points are nearly co-circular.

Another advantage of an arbitrary triangulation is that the predicates are cheaper to compute than for Delaunay. For polynomial trajectories of degree d, the failure times of the *Orientation* predicates are roots of polynomials of degree $2d$, against degree $4d$ for the *InCircle* predicates.

Note that the lazy strategy may not be efficient on the long term, as the total number of events may be higher than with a more foreseeing strategy like Delaunay. However, the actual motivation for this choice is not efficiency, but fidelity to deformations and topology changes of the interfaces. The lazy behavior guarantees that the triangulation remains a faithful embedding of the polygonal interfaces. A connectivity change for the convenience of the kinetic data structure would constitute an unnecessary perturbation of the location of the interfaces.

An objection to the lazy kinetic triangulation could be the absence of control on the quality of the triangles. In some other contexts where the numerical robustness directly depends on the quality of the worst-shaped element of the triangulation, this can impair the simulation. This limitation does not apply to our approach, because we are only interested in the interfaces between the different materials. The tiling of each material is not used in the simulation, it is only available internally to the kinetic data structure.

3.3. Predicates and motion model

The convex hull can be kinetically maintained by considering an orientation predicate for each triplet of consecutive border vertices. However, for the sake of simplicity, we assume here that the convex hull remains constant during motion. This can be easily achieved by adding to the triangulation fixed points that delimit a bounding box of the simulation. As a result, the certification of our lazy kinetic triangulation only requires one orientation predicate for each triangle.

As for the motion model of the vertices between two consecutive steps of the simulation, it can be chosen arbitrarily with a canonically-defined structure such as the Delaunay triangulation, as long as it respects the continuity of the trajectories. In this case, indeed, the intermediates states of the triangulation can be safely disregarded. In [GR04], Guibas and Russel empirically compare the efficiency of three different motion models for the 3D kinetic Delaunay triangulation: a linear motion of all the points simultaneously, a linear motion of one point at a time, and a linear motion of all the points along one coordinate at a time.

In contrast, the state of our lazy kinetic triangulation depends on the history of the motion, and the choice of the motion model conditions the fidelity to the simulated physical phenomenon. Thus, the "one point at a time" and "one coordinate at a time" motion models would be questionable, despite their higher efficiency. In most applications, linear or quadratic trajectories are relevant. Quadratic trajectories faithfully model accelerated motion, but are significantly more expensive computationally, because they require to manipulate roots of polynomials of degree 4.

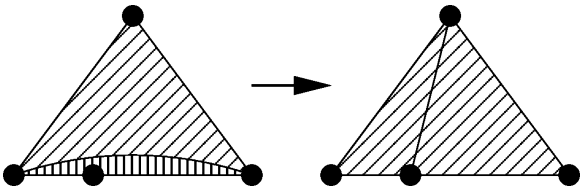

Figure 2: *Example of the removal of a flat triangle by flipping its longest edge. The fill pattern represents the labels of the triangles.*

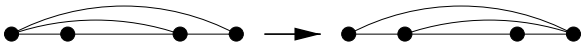

Figure 3: *Example of an ineffective edge flip: it replaces two flat triangles with two new flat triangles in a similar configuration.*

Figure 4: *Example of an acceptable edge flip: it replaces two flat triangles with two new flat triangles, but strictly decreases the length of the possible flipping edges.*

3.4. Processing of the events

First, we consider the generic event of a flat triangle whose vertices have three distinct positions. We use the classical Lawson *edge flip* [Law77] to restore a valid geometric triangulation. The longest edge of the flat triangle is flipped, as shown in Figure 2. An edge flip requires the two adjacent triangles to form a convex quad. Note that this condition is automatically fulfilled in our case, since three vertices are collinear.

Also, consistent labels must be assigned to the two new triangles generated by the edge flip: these two triangles are given the label of the original non-flat triangle involved in the flip (*cf* Figure 2), so that the location of the interfaces is not altered. Finally, the predicates of these two new triangles are built and their failure times are placed in the priority queue.

As a result of these changes in the connectivity and in the labels, some vertices of the triangulation may no longer participate in the interface, i.e., all the adjacent triangles may now have the same label. It typically happens in the case of a topology change of the materials or of a shock, as we will show in our experiments in Section 4. For the sake of efficiency, these vertices are removed from the triangulation and their holes are retriangulated, again without affecting the position of the interfaces. There is no ambiguity in assigning labels to the newly created triangles.

We now describe how to cope with degenerate cases, in order to make our algorithm perfectly reliable. Flat triangles with two or three vertices having the same position are handled with the *edge collapse* operation. It consists in collapsing the two vertices of an edge into a single vertex. One of the vertices of the edge, the edge itself and the two triangles adjacent to the edge are deleted. In the edges and triangles involving the deleted vertex, the latter is replaced by the other vertex of the collapsed edge. In general, an edge collapse may generate triangles of reverse orientation, and make a geometric triangulation invalid. Note that it is not possible in our case, since the vertices of the collapsed edge have the same position. As regards the labels of the triangles, no change is needed after an edge collapse.

The handling of simultaneous events also requires special care. It happens when several triangles become flat exactly at the same time. The kinetic data structures framework does not prescribe the order in which simultaneous events must be processed. This order depends on the particular attribute maintained, and turns out to be important for our lazy kinetic triangulation. Indeed, processing the events in a wrong order can compromise the termination of the algorithm.

This is due to the fact that, in some degenerate cases, the processing of an event can generate one of several new flat triangles. For example, it happens when an edge between two flat triangles is flipped. We give an example of such a situation in Figure 3. There are two possible edge flips for these two adjacent flat triangles. If the shortest of the two possible flipping edges is flipped first, two new flat triangles are created, and the final configuration is qualitatively the same as before the flip. If the algorithm constantly chooses the wrong flipping edge, it enters an infinite loop.

In contrast, in Figure 4 we show a configuration in which an edge flip between two flat triangles makes some progress. The difference between these two degenerate configurations inspires the following strategy to handle simultaneous events. We perform in priority any needed edge collapse. Then, we process any needed edge flip, except that *we skip the flipping edges adjacent to two flat triangles if the flipping edge is not the largest edge of both triangles*. After an edge flip, new edge collapses may become possible: we perform them before considering the next edge flip.

Now, we give an elementary proof, in three arguments, that the algorithm terminates with this strategy, and hence that it restores a valid geometric triangulation. First, it is trivial that the sequence of edge collapses terminates, since each edge collapse deletes a vertex. Second, among the possible edge flips, there is always an acceptable one, for instance the one with the largest flipping edge. Third, for each acceptable edge flip:

- either it strictly decreases the number of flat triangles and of flipping edges, as shown in Figure 2,
- or it strictly decreases the length of the possible flipping edges, while keeping the number of flat triangles constant, as illustrated in Figure 4.

Throughout this section, we have assumed that the convex hull remains constant. However, handling specifically the events involving border vertices and border edges suffices to extend the above algorithm to the case of a varying convex hull.

3.5. Resampling of the interfaces

In most applications, the resolution of the interfaces has to be corrected during deformation, to maintain the desired localization accuracy. Although this task is highly application-dependent, we propose a basic technique to control the distribution of the lengths of interface edges. This resampling step is performed at the end of each step of the simulation, after a valid geometric triangulation has been restored by the kinetic data structure.

Basically, the edges of the interfaces, hence not all the edges of the embedding triangulation, are tested against a maximum and a minimum threshold. Long edges are bisected by adding their center point to the triangulation, then replacing each of the triangle adjacent to the edge by two new triangles with identical labels. Note that this modification neither changes the location of the interface, nor jeopardizes the validity of the triangulation.

In contrast, collapsing short edges requires special care, because it alters the position of the interface and it can sometimes generate inverted triangles. We emphasize that, although very short edges are an unnecessary computational burden, they do not affect the stability of the simulation, as our approach is not subject to the formation of swallowtails.

4. Numerical Experiments

4.1. Implementation issues

By using CGAL (Computational Geometry Algorithms Library, homepage: www.cgal.org) [BDTY00], we have been able to implement our approach with only 1500 lines of C++ code. We have taken advantage of the flexibility of this library to derive our labeled triangulation data structure from the existing triangulation data structure. In particular, we have overloaded the edge flip and edge collapse operations to consistently handle the labels of the triangles.

We have developed our own implementation of the kinetic data structure. Our priority queue is implemented with a *pairing heap* [FSST86], an efficient self-adjusting heap data structure. In the future, we will probably consider conforming to the CGAL package implementing the computational framework of Guibas, Karavelas and Russel [GKR04].

In order to make our implementation robust, the exact computation paradigm [Yap97] can be followed. The coordinates of the vertices may be represented with arbitrary precision rational numbers provided by the GMP (GNU Multiple Precision) arithmetic library [GMP]. In this case, the failure times of the predicates are the roots of polynomials with rational coefficients. The method of Emiris and Tsigaridas [ET04], based on Sturm sequences, may be used to sort them in the priority queue. The latter method also allows to exactly determine the sign of a polynomial at the root of another polynomial. This can be used to robustly compare the lengths of the different edges of a flat triangle, to choose what edge collapse and what edge flip must be performed.

However, in general, the exact computation paradigm does not apply to the numerical solver. Consequently, in our implementation, for efficiency purposes, we have used a fixed-precision floating-point arithmetic both for the solver and for the lazy kinetic triangulation. This avoids converting the coordinates of the vertices back and forth between floats and exact number representations. We have not encountered any robustness issues in our numerical experiments.

4.2. Mass loss

In our first numerical experiment, we compare the behavior of the level set method and of our method regarding mass loss, using an experimental methodology initially proposed by LeVeque [LeV96], and later popularized in computer graphics by Enright and coworkers [EFFM02; EMF02; ELF05].

The spatial domain is the unit square $[0, 1] \times [0, 1]$, the initial interface is a circle with radius 0.15 centered at (0.5, 0.75) and the evolution is driven by an analytical incompressible velocity field with non-constant vorticity that varies sinusoidally in time with a period T. Specifically, the velocity field $\boldsymbol{v}$ is defined by

$$\boldsymbol{v}(x, y, t) = \cos\frac{\pi t}{T}\begin{pmatrix}\sin^2(\pi x)\sin(2\pi y)\\ -\sin(2\pi x)\sin^2(\pi y)\end{pmatrix}. \quad (1)$$

This experiment is challenging because the flow considerably stretches the interface, as can be seen in Figure 5. Note that the exact solution of this evolution is not available at all time, so at first sight we cannot measure the error. But the velocity reverses at time $T/2$, so that the initial interface should be recovered at time T. This provides a convenient way to evaluate the mass loss caused by each method. In this experiment, we take $T = 4$.

Due to the gap between Lagrangian and Eulerian viewpoints, a direct comparison between the two methods for the same parameter values is not possible. We have done our best

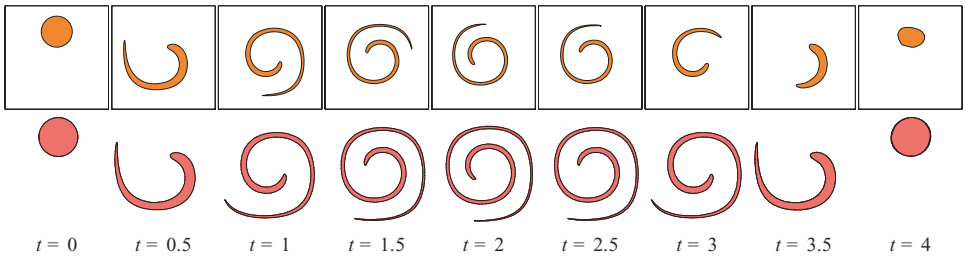

Figure 5: *Comparison between the level set method (top) and our method (bottom) for a circle stretched and distorted by a "vortex-in-a-box" velocity field.*

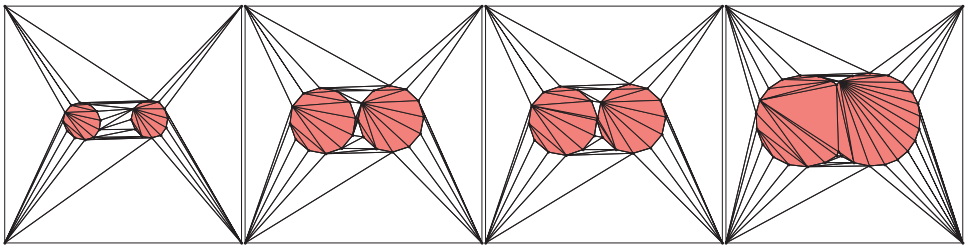

Figure 6: *From left to right, different stages of the coalescence of two expanding objects.*

in adjusting these parameters so as to obtain the fairest possible comparison. In both experiments, we use a simple Euler time stepping scheme with a time step $dt = 0.05/128$. We use a highly-optimized home-made implementation of the level set method used in our previous work [PHKF06], with a 128×128 resolution. To fix ideas, extracting the isocontour from the initial level set function generates 156 vertices. We set to 100 the initial number of vertices in our kinetic triangulation.

Figure 5 displays the evolution obtained with the level set method (top) and with our method (bottom) at different times. As expected, the level set method fails to preserve the thin parts of the interface, despite a quite high resolution and the use of a high-order spatial differencing scheme (namely a fifth-order WENO scheme [JS96, JP00]). As a result, a significantly higher mass loss is observed with the level set method.

4.3. Example of coalescence

In our second experiment, we demonstrate how our approach handles topology changes, in the simple case of the coalescence of two expanding objects. Here, the number of vertices is very low in order to ease the visualization of the embedding triangulation. In this simulation, the numerical solver only advects the vertices with a unit speed in the direction of the estimated normal. To be more precise, we obtain this velocity field by a *variational* paradigm, as the *gradient ascent* of the area enclosed by the interfaces.

Figure 6 shows different stages of the evolution. We invite the reader to have a close look at the connectivity and the labels of the embedded triangulation just before and after the two objects merge.

4.4. Three-phase combustion simulation

In our third experiment, we simulate the combustion of two different materials plunged in a third ambient material, to demonstrate the ability of our approach to accommodate any number of materials, and in particular multiple junctions. This type of motion generates complex phenomena called *shocks* and *rarefactions* in the study of conservation laws [LeV92]. Whereas shocks are extremely difficult to simulate with a traditional front-tracking method, due to the repeated formation of swallowtails, they are handled efficiently and robustly by our approach.

Figure 7 shows this simulation at various time samples. The combustion speed of the blue material is four times that of the red material, which creates a complex and interesting series of topology changes. In this figure, the tiling inside the materials is not represented, due to the large number of vertices. Anyway, we again emphasize that this tiling is only relevant to the internal functioning of the kinetic lazy triangulation.

4.5. Brain segmentation from medical images

In our last experiment, we give a glimpse of the possible applications of our approach in medical imaging.

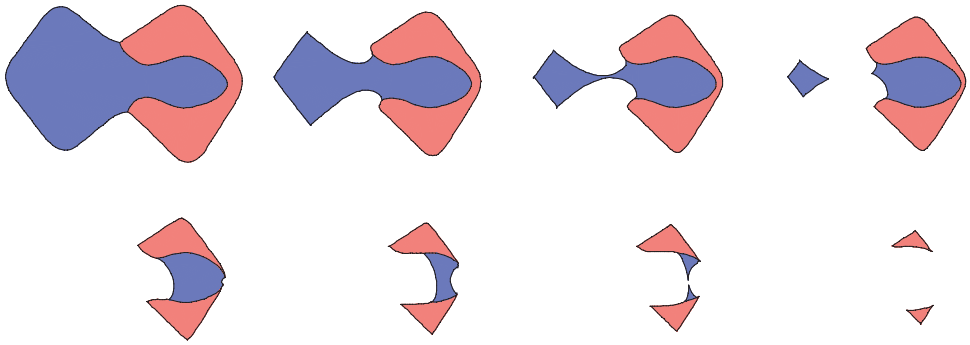

Figure 7: *(From left to right and top to bottom, different stages of a three-phase combustion simulation.*

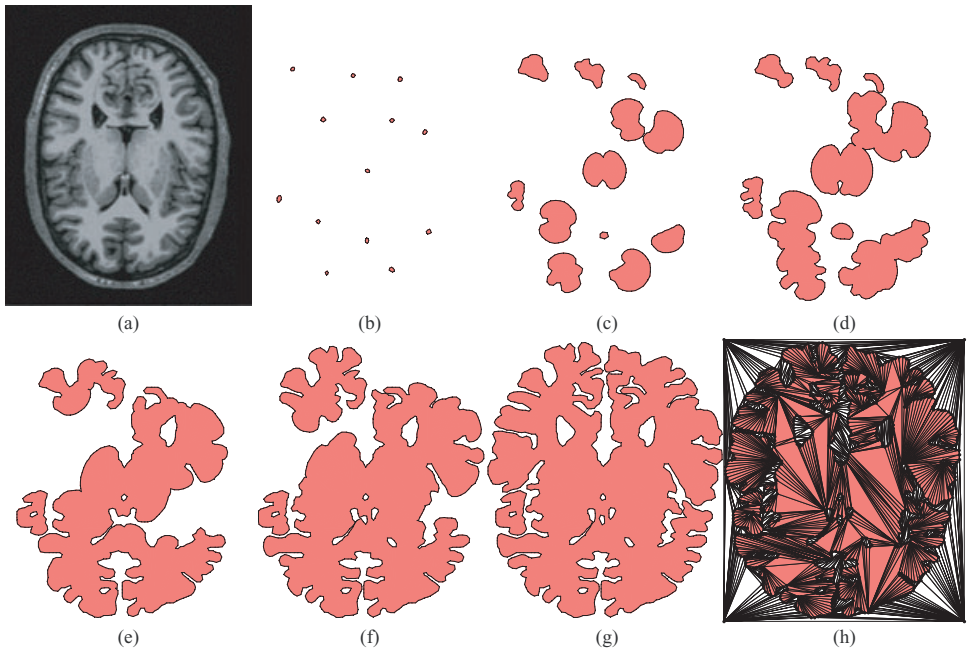

Figure 8: *Application to brain segmentation: (a) MR image, (b-g) different stages of the evolution, (h) labeled triangulation of the final shape.*

Deformable curves and surfaces are a widely used technique for shape reconstruction in image processing. We refer the interested reader to two reviews on this topic [MT96, XPP00].

Here, we address the automatic delineation of the brain from magnetic resonance imaging (MRI) of the head. Figure 8 shows the 2D magnetic resonance image whose intensities drive the motion, the different stages of the evolution and the embedding triangulation of the final interface. Note how the initial seeds grow and progressively merge to fit the complex shape of the brain.

4.6. Quantitative evaluation

Table 1 gathers some quantitative measurements on our different experiments:

- the number of vertices, at the beginning and at the end of the evolution,
- the total number of events,
- the minimum, maximum and average computation time required by one iteration of our algorithm. The time spent in the numerical solver and the time needed to update

Table 1: *Number of vertices, total number of events and measurements of the computation time in our different experiments.*

				Comput. time / it. (ms)					
	# Vertices			Solver			Kinetic triangulation		
Experiment	Initial	Final	# Events	min.	ave.	max.	min.	ave.	max.
Merge	28	51	26	0.02	0.09	0.28	0.03	0.06	0.12
Combustion	300	4	3276	0.03	0.12	0.34	0.02	0.30	0.54
Brain	116	1180	4148	0.2	1.5	2.6	0.02	0.9	1.7

our lazy kinetic triangulation are given separately. These measurements were made on a 2.8 GHz Intel Xeon workstation running under Windows XP.

Although the numerical solvers used in our experiments are particularly simple and computationally inexpensive, the computation time required to maintain our lazy kinetic triangulation keeps lower. Our more complex simulation, brain segmentation, runs in about 300 milliseconds on a standard workstation.

5. Discussion and Future Work

We have proposed a robust and efficient Lagrangian approach for modeling dynamic interfaces between different materials undergoing large deformations. Thanks to the kinetic data structures framework and the exact computation paradigm, the reliability of our algorithm, even in difficult situations such as shocks and topology changes, can be certified. Being purely Lagrangian, our method does not suffer from mass loss which plagues the level set method, and can track material properties such as color or texture coordinates during motion at no additional cost, while being free of the localization problem of particle-based approaches. Whereas our work in its current progress does not claim state-of-the-art results, we have demonstrated in a series of numerical experiments that it brings a promising rigorous alternative to existing techniques for two-dimensional and axisymmetric simulations.

We now discuss the feasibility of an extension of our approach to three dimensions. As for the embedding triangulation, it immediately extends to a tetrahedralization, whose cells are labeled with the type of material they contain. All the advantages of this representation still hold, such as the automatic enforcement of watertight surfaces and the seamless handling of multiple junctions.

However, as expected, maintaining this tetrahedralization is significantly more complex than in two dimensions. First, the predicates needed to certify the kinetic lazy triangulation are one orientation predicate for each tetrahedron. It requires to manipulate polynomials of degree 3, assuming a linear motion model, while a quadratic motion model now becomes prohibitively expensive.

But actually, the main difficulty is that more complex events must be processed to faithfully track the location of the interfaces. In particular, the addition of Steiner points cannot be avoided in some situations, which is consistent with the well known fact that many polyhedra have no tetrahedralization [RS92]. Thus, we cannot handle some events with the classical edge collapse, $2 \rightarrow 3$ and $3 \rightarrow 2$ tetrahedra flips. For instance, when two opposite edges of a flat tetrahedron intersect, it is sometimes mandatory to add the intersection point to the triangulation. Our future work includes investigating whether the number of these "spontaneous" Steiner points becomes prohibitive for the simulation.

References

[ABdB*99] Agarwal P., Basch J., De Berg M., Guibas L. and Hershberger J. Lower bounds for kinetic planar subdivisions. In *Annual Symposium on Computational Geometry* (1999), pp. 247–254.

[ABG*02] Agarwal P., Basch J., Guibas L., Hershberger J. and Zhang L. Deformable free-space tilings for kinetic collision detection. *The International Journal of Robotics Research 21*, 3 (2002), 179–197.

[ABL03] Attali D., Boissonnat J.-D. and Lieutier A. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *Annual Symposium on Computational Geometry* (2003), pp. 201–210.

[AGE*02] Agarwal P. K., Guibas L. J., Edelsbrunner H., Erickson J., Isard M., Har-Peled S., Hershberger J., Jensen C., Kavraki L. E., Koehl P., Lin M., Manocha D., Metaxas D., Mirtich B., Mount D., Muthukrishnan S., Pai D., Sacks E., Snoeyink J., Suri S. and Wolfson O. Algorithmic issues in modeling motion. *ACM Computing Surveys 34*, 4 (2002), 550–572.

[AGMR98] Albers G., Guibas L., Mitchell J. and Roos T. Voronoi diagrams of moving points. *International Journal of Computational Geometry and Applications 8*, 3 (1998), 365–380.

[AS95] Adalsteinsson D. and Sethian J. A fast level set method for propagating interfaces. *Journal of Computational Physics 118*, 2 (1995), 269–277.

[AWY04] Agarwal P., Wang Y. and Yu H. A 2D kinetic triangulation with near-quadratic topological changes. In *Annual Symposium on Computational Geometry* (2004), pp. 180–189.

[BDTY00] Boissonnat J.-D., Devillers O., Teillaud M. and Yvinec M. Triangulations in CGAL. In *Annual Symposium on Computational Geometry* (2000), pp. 11–18.

[BEG*99] Basch J., Erickson J., Guibas L., Hershberger J. and Zhang L. Kinetic collision detection between two simple polygons. In *Symposium on Discrete Algorithms* (1999), pp. 102–111.

[BGH99] Basch J., Guibas L. and Hershberger J. Data structures for mobile data. *Journal of Algorithms 31*, 1 (1999), 1–28.

[BGOS06] Bargteil A., Goktekin T., O'Brien J. and Strain J. A semi-Lagrangian contouring method for fluid simulation. *ACM Transaction on Graphics 25*, 1 (2006), 19–38.

[CD97] Cani M.-P. and Desbrun M. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics 3*, 1 (1997), 39–50.

[COQ06] Cecil T., Osher S. and Qian J.-L. Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. *Journal of Computational Physics 213*, 2 (2006), 458–473.

[DC96] Desbrun M. Cani M.-P. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation* (1996), pp. 61–76.

[DG95] Desbrun M. and Gascuel M.-P. Animating soft substances with implicit surfaces. In *Proceedings of ACM SIGGRAPH* (1995), pp. 287–290.

[EFFM02] Enright D., Fedkiw R., Ferziger J. and Mitchell I. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 1 (2002), 83–116.

[ELF05] Enright D., Losasso F. and Fedkiw R. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures 83* (2005), 479–490.

[EMF02] Enright D., Marschner S. and Fedkiw R. Animation and rendering of complex water surfaces. In *Proceedings of ACM SIGGRAPH* (2002), pp. 736–744.

[ET04] Emiris I. and Tsigaridas E. Comparing real algebraic numbers of small degree. In *European Symposium on Algorithms* (2004), pp. 652–663.

[FF01] Foster N. and Fedkiw R. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH* (2001), pp. 23–30.

[FM96] Foster N. and Metaxas D. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483.

[FSST86] Fredman M., Sedgewick R., Sleator D. Tarjan R. The pairing heap: A new form of self-adjusting heap. *Algorithmica 1*, 1 (1986), 111–129.

[GKR04] Guibas L., Karavelas M. and Russel D. A computational framework for handling motion. In *Workshop on Algorithm Engineering and Experiments* (2004), pp. 129–141.

[GMP] GMP, the GNU multiple precision arithmetic library. Homepage: http://www.swox.com/gmp/.

[GR04] Guibas L. and Russel D. An empirical comparison of techniques for updating Delaunay triangulations. In *Annual Symposium on Computational Geometry* (2004), pp. 170–179.

[Gui04] Guibas L. Modeling motion. In *Handbook of Discrete and Computational Geometry*,Goodman J., O'Rourke J., (Eds.). Chapman and Hall/CRC, (2004), pp. 1117–1134.

[HN81] Hirt C. and Nichols B. Volume of fluid ({VOF}) method for the dynamics of free boundaries. *Journal of Computational Physics 39*, 1 (1981), 201–225.

[JP00] Jiang G.-S. and Peng D. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM Journal of Scientific Computing 21*, 6 (2000), 2126–2143.

[JS96] Jiang G.-S. and Shu C.-W. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics 126*, 1 (1996), 202–228.

[KFCO06] Klingner B., Feldman B., Chentanez N. and O'Brien J. Fluid animation with dynamic meshes, (2006).

[KS02] Kirkpatrick D. and Speckmann B. Kinetic maintenance of context-sensitive hierarchical representations for disjoint simple polygons. In *Annual Symposium on Computational Geometry* (2002), pp. 179–188.

[Law77] Lawson C. Software for C^1 surface interpolation. In *Mathematical Software III*,Rice J., (Ed.). Academic Press, New York, (1977), pp. 161–194.

[LeV92] LeVeque R. *Numerical M*ethods for Conservation Laws. Birkhäuser, Basel, (1992).

[LeV96] LeVeque R. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal of Numerical Analysis 33*, 2 (1996), 627–665.

[LFO06] Losasso F., Fedkiw R. and Osher S. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids* (2006). To appear.

[LGF04] Losasso F., Gibou F. and Fedkiw R. Simulating water and smoke with an octree data structure. In *Proceedings of ACM SIGGRAPH* (2004), pp. 457–462.

[LM99] Lachaud J.-O. and Montanvert A. Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction. *Medical Image Analysis 3*, 2 (1999), 187–207.

[LT05] Lachaud J.-O. and Taton B. Deformable model with a complexity independent from image resolution. *Computer Vision and Image Understanding 99*, 3 (2005), 453–475.

[MCG03] Müller M., Charypar D. and Gross M. Particle-based fluid simulation for interactive applications. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159.

[MKN*04] Müller M., Keiser R., Nealen A., Pauly M., Gross M. and Alexa M. Point based animation of elastic, plastic and melting objects. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 141–151.

[MT96] McInerney T. Terzopoulos D. Deformable models in medical image analysis: a survey. *Medical Image Analysis 1*, 2 (1996), 91–108.

[MT99] McInerney T. Terzopoulos D. Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Transactions on Medical Imaging 18*, 10 (1999), 840–850.

[MT00] McInerney T. and Terzopoulos D. T-snakes: Topology adaptive snakes. *Medical Image Analysis 4*, 2 (2000), 73–91.

[OF01] Osher S. and Fedkiw R. Level set methods: an overview and some recent results. *Journal of Computational Physics 169*, 2 (2001), 463–502.

[OS88] Osher S. and Sethian J. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics 79*, 1 (1988), 12–49.

[PHKF06] Pons J., Hermosillo G., Keriven R. and Faugeras O. Maintaining the point correspondence in the level set framework. *Journal of Computational Physics* (2006). To appear.

[PKA*05] Pauly M., Keiser R., Adams B., Dutré P., Gross M. and Guibas L. Meshless animation of fracturing solids. *ACM Transactions on Graphics 24*, 3 (2005), 957–964.

[PMO*99] Peng D., Merriman B., Osher S., Zhao H.-K. and Kang M. A PDE-based fast local level set method. *Journal of Computational Physics 155*, 2 (1999), 410–438.

[PTB*03] Premože S., Tasdizen T., Bigler J., Lefohn A. and Whitaker R. Particle-based simulation of fluids. *Computer Graphics Forum 22*, 3 (2003), 401–410.

[RS92] Ruppert J. and Seidel R. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry 7*, 3 (1992), 227–253.

[Set99] Sethian J. *Level S*et Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press, (1999).

[SJ02] Shin S. and Juric D. Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. *Journal of Computational Physics 180*, 2 (2002), 427–470.

[SSC02] Smith K., Solis F. and Chopp D. A projection method for motion of triple junctions by level sets. *Interfaces and Free Boundaries 4*, 3 (2002), 263–276.

[SSO94] Sussman M., Smereka P. and Osher S. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics 114*, 1 (1994), 146–159.

[Str01] Strain J. A fast semi-Lagrangian method for moving interfaces. *Journal of Computational Physics 169*, 1 (2001), 1–22.

[TBE*01] Tryggvason G., Bunner B., Esmaeeli A., Juric D., Al-Rawahi N., Tauber W., Han J., Nas S. and Jan Y.-J. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics 169*, 2 (2001), 708–759.

[TPF89] Terzopoulos D., Platt J. and Fleischer K. Heating and melting deformable models (from goop to glop). *Graphics Interface* (1989), 219–216.

[UT92] Unverdi S. and Tryggvason G. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics 100*, 1 (1992), 25–37.

[XPP00] Xu C., Pham D. and Prince J. Medical image segmentation using deformable models. In *Handbook of Medical Imaging*,Fitzpatrick J., Sonka M., (Eds.), vol. 1. SPIE Press, (2000), ch. 3, pp. 129–174.

[Yap97] Yap C.-K. Towards exact geometric computation. *Computational Geometry: Theory and Applications 7*, 1-2 (1997), 3–23.